A.1 Kurzanleitung zum Erstellen eines Messprogramms in LabVIEW

Die vorliegende Kurzanleitung enthält die Voraussetzungen und die grundlegenden Kenntnisse zum Programmieren in LabVIEW, die für den Versuch „Lange Halbwertszeiten“ erforderlich sind. Sie basiert auf dem deutschsprachigen Benutzerhandbuch von LabVIEW [14] zur Vollversion LabVIEW 7.0, aus der Textauszüge und Abbildungen übernommen wurden.
Mit Hilfe dieser Kurzanleitung wird es den Studenten ermöglicht, auch ohne Vorkenntnisse des Programmierens in LAbVIEW das Versuchsziel in der vorgesehenen Zeit erfüllen zu können.

Kapitel 1 „Virtuelle Instrumente“ und Kapitel 2 „LabVIEW Umgebung“ geben eine allgemeine Einleitung in den Aufbau und die Struktur von LabVIEW. Kapitel 3 „Erstellen des Frontpanels“ und Kapitel 4 „Erstellen des Blockdiagramms“ führen die wesentlichen Kenntnisse zur Anfertigung der Benutzeroberflächen auf. In den übrigen Kapiteln 5 bis 9 werden konkrete Programmstrukturen sowie das Verarbeiten, Darstellen und Speichern von Daten diskutiert.

Bei weiteren Fragen wird auf das genannte Benutzerhandbuch [14] verwiesen.

Kapitel 1: Virtuelle Instrumente

Die LabVIEW-Programme werden als virtuelle Instrumente (VIs) bezeichnet, da mit Erscheinungsbild und Funktion physische Instrumente wie beispielsweise Oszilloskope und Multimeter nachgebildet werden können. Jedes VI arbeitet mit Funktionen, die Eingaben von der Benutzeroberfläche oder aus anderen Quellen verarbeiten. Diese Informationen können dann angezeigt werden oder in andere Dateien oder auf andere Computer verschoben werden.

Ein VI enthält drei Komponenten:
Das Frontpanel, das Blockdiagramm und das Symbol- und Anschlussfeld.

Frontpanel
Das Frontpanel ist die Benutzeroberfläche des VIs. Abbildung A.1 zeigt ein Beispiel für ein Frontpanel des Versuchs „Lange Halbwertszeiten“:

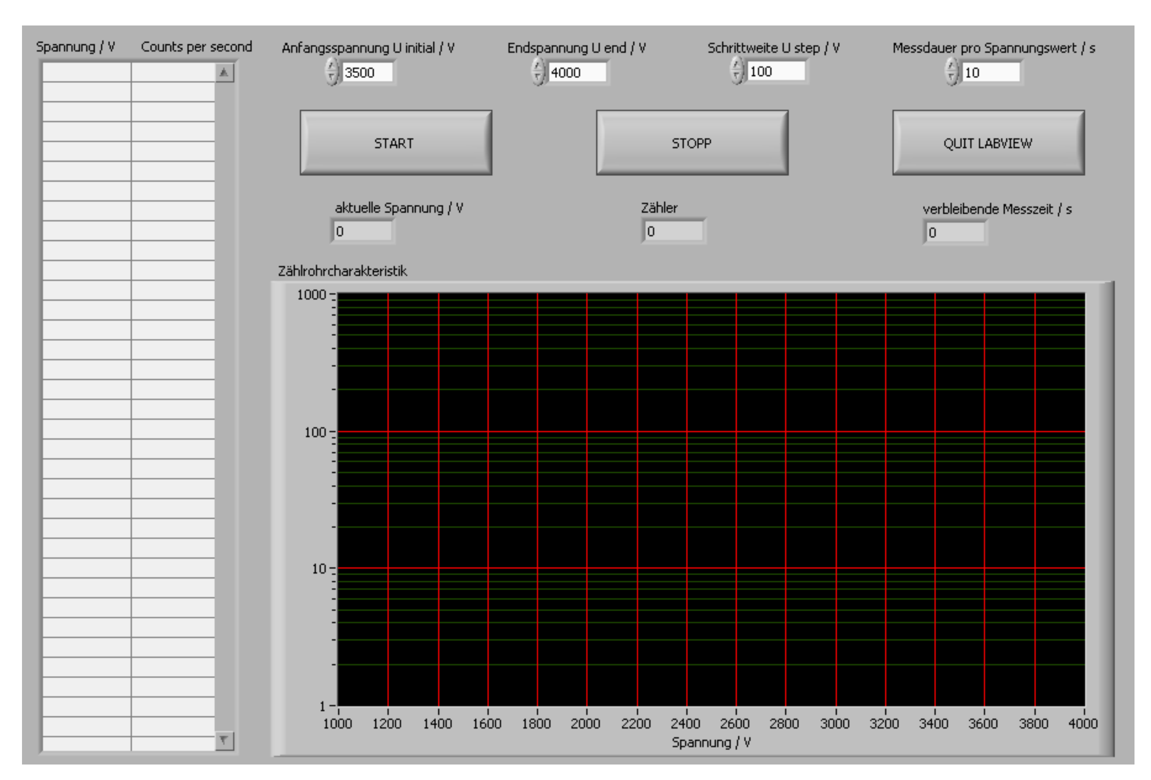


Abbildung A.1: Beispiel eines Frontpanels zum Versuch „Lange Halbwertszeiten“

Das Frontpanel wird mit Bedien- und Anzeigeelementen erstellt, welche die interaktiven Ein- bzw. Ausgabeanschlüsse des VIs darstellen. Bedienelemente sind Knöpfe, Drehregler und andere Eingabeelemente, mit denen Eingabegeräte simuliert und Daten an das Blockdiagramm des VIs übergeben werden. Anzeigeelemente sind

Graphen, LEDs und sonstige Anzeigen, mit denen Ausgabegeräte nachgeahmt und die Daten angezeigt werden, die vom Blockdiagramm erfasst oder erzeugt werden.

Blockdiagramm

Nach der Erstellung des Frontpanel können mit Hilfe grafisch dargestellter Funktionen Code hinzugefügt werden, um die Frontpanel-Objekte zu steuern. Das Blockdiagramm enthält dann diesen grafischen Quellcode. Frontpanel-Objekte werden im Blockdiagramm als Anschluss-Terminals dargestellt. In Abbildung A.2 ist ein einfaches Blockdiagramm und das dazugehörige Frontpanel dargestellt.

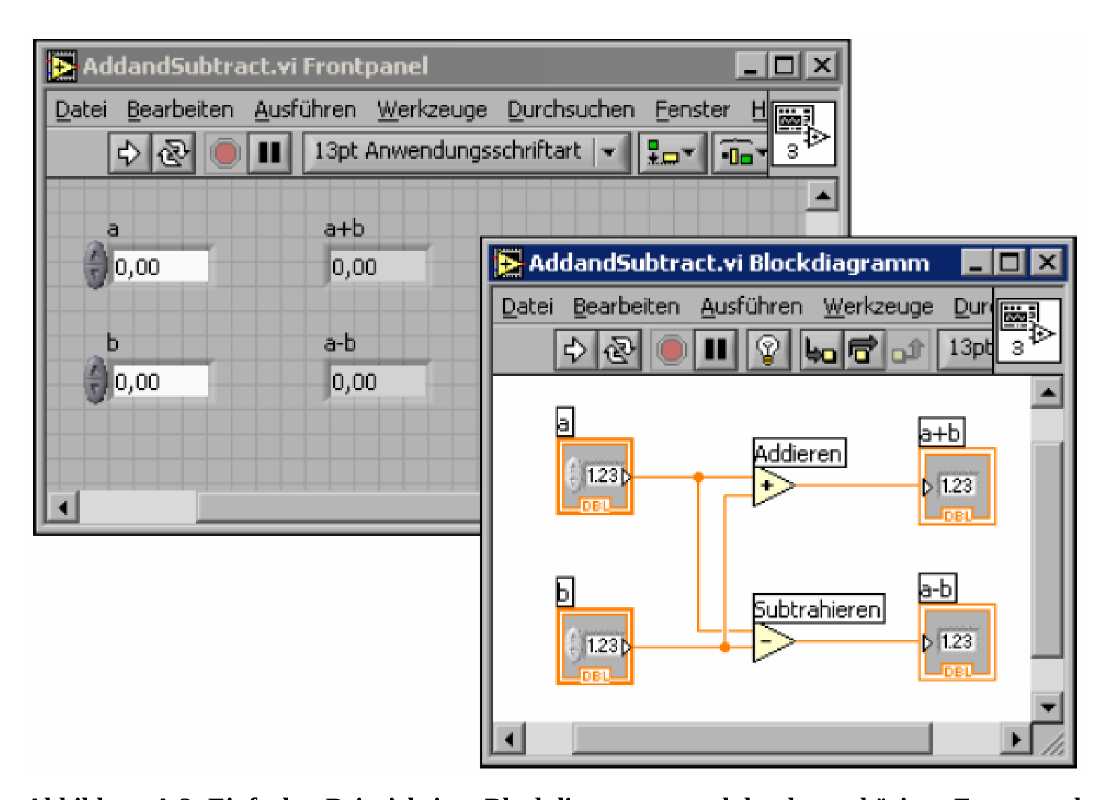


Abbildung A.2: Einfaches Beispiel eines Blockdiagramms und des dazugehörigen Frontpanels

Symbol- und Anschlussfeld

Zu jedem VI gehört ein Symbolfeld, das in der rechten oberen Ecke des Frontpanels und des Blockdiagramms angezeigt wird. Es dient zur grafischen Darstellung eines VIs und dient als knappe Beschreibung des vorliegenden VIs. Es lässt sich zudem als Anschlussfeld darstellen, in dem die Eingänge und Ausgänge definiert werden, die mit dem VI verbunden werden sollen. Es werden Verbindungen definiert, indem jedem Anschluss im Anschlussfeld ein Bedien- oder Anzeigeelement auf dem Frontpanel zugewiesen wird. Durch Verbinden der Anschlüsse kann das vorliegende VI als SubVI (Unterprogramm) eingesetzt werden.

Anschlüsse, Funktionen und Verbindungen

Anschlüsse

Die Symbole der Anschlüsse verweisen auf den Datentyp des Bedien- oder Anzeigeelements. Die Elemente des Frontpanels können so konfiguriert werden, dass sie im Blockdiagramm entweder als Symbol oder als Anschluss eines bestimmten Datentyps dargestellt werden. So wird zum Beispiel mit einem Drehknopfsymbol ein auf dem Blockdiagramm befindlicher Drehknopf dargestellt (Abbildung A.3b). Das "DBL" am unteren Rand zeigt den Datentyp an und bedeutet, dass dieses Element arbeitet mit Fließkommazahlen mit doppelter Genauigkeit.

Dagegen ist bei Darstellung als DBL-Symbol (Abbildung A.3c) nur ersichtlich, dass es sich um ein numerisches Bedien- oder Anzeigeelement handelt, das diesen Datentyp verwendet.



Abbildung A.3: Darstellung eines Drehknopfsymbols
a) auf dem Frontpanel
b) als Drehknopf auf dem Blockdiagramm
c) als Datentypsymbol

Anschlüsse sind Eingangs- und Ausgangsports, über die Informationen zwischen dem Frontpanel und dem Blockdiagramm ausgetauscht werden. Daten, die über die Bedienelemente des Frontpanels eingegeben werden, werden über die Bedienelement-terminals an das Blockdiagramm übergeben (Abbildung A.2). Anschließend passieren sie die Additions- und Subtraktionsfunktionen. Wenn die Additions- und Subtraktionsfunktionen die internen Berechnungen abgeschlossen haben, werden neue Datenwerte erstellt. Diese fließen zu den Anzeigeelementanschlüssen, wo sie das Blockdiagramm verlassen, um erneut an das Frontpanel übergeben und dort angezeigt zu werden.

Beziehung zwischen Frontpanelobjekten und Blockdiagrammanschlüssen

Frontpanelobjekte werden im Blockdiagramm als Anschlüsse dargestellt. Wenn man wissen möchte, für welches Frontpanelelement ein bestimmtes Blockdiagrammobjekt steht, klickt man dieses doppelt an. Das entsprechende Element wird dann im Frontpanel hervorgehoben.

Knoten

Knoten sind Objekte im Blockdiagramm, die über Eingänge und/oder Ausgänge verfügen und Funktionen in einem laufenden VI ausführen. Knoten wie z.B. Additions- und Subtraktionsfunktionen in Abbildung A.2 entsprechen Anweisungen, Operatoren, Funktionen und Subroutinen in textbasierten Programmiersprachen.

Verbindungen

Sie übertragen die Daten über Verbindungsleitungen zwischen den Blockdiagrammobjekten. (In Abbildung A.2 werden die als Bedien- und Anzeigeelemente fungierenden DBL-Anschlüsse über Verbindungsleitungen mit den Additions- und Subtraktionsfunktionen verbunden.) Jede Verbindung verfügt über eine einzige Datenquelle, die sie jedoch mit mehreren Daten lesenden VIs und Funktionen verbinden können. Verbindungen weisen in Abhängigkeit ihres Datentyps unterschiedliche Farben, Stile und Stärken auf. Eine unterbrochene Verbindung wird als eine gestrichelte schwarze Linie mit einem roten X in der Mitte dargestellt.

Strukturen

Strukturen sind grafische Darstellungen der Schleifen und Case-Anweisungen in textbasierten Programmiersprachen. Man verwendet Strukturen im Blockdiagramm, um Codeblöcke zu wiederholen und Code bedingungsabhängig oder in einer bestimmten Reihenfolge auszuführen (siehe Kapitel 6).

Kapitel 2: Die LabVIEW Umgebung

Zum Erstellen der Frontpanels und Blockdiagramme von VIs stehen Paletten, Werkzeuge und Menüs von LabVIEW zur Verfügung.

Elementepalette

Die Palette „Elemente“ steht nur auf dem Frontpanel zur Verfügung. Sie enthält die Bedien- und Anzeigeelemente zur Erstellung der Benutzeroberfläche eines VIs. Je nach ausgewählter Palettenansicht können unterschiedliche Bedien- und Anzeigeelemente zu sehen sein (Abbildung A.4).

Die Elementepalette kann entweder durch Klicken auf „Fenster » Elementepalette“ oder mit der rechten Maustaste auf dem Arbeitsbereich des Frontpanels angezeigt und beliebig auf dem Bildschirm verschoben werden. Die aktuelle Position der Palette wird beim Schließen von LabVIEW gespeichert, so dass sie beim nächsten Start wieder an derselben Stelle angezeigt wird.

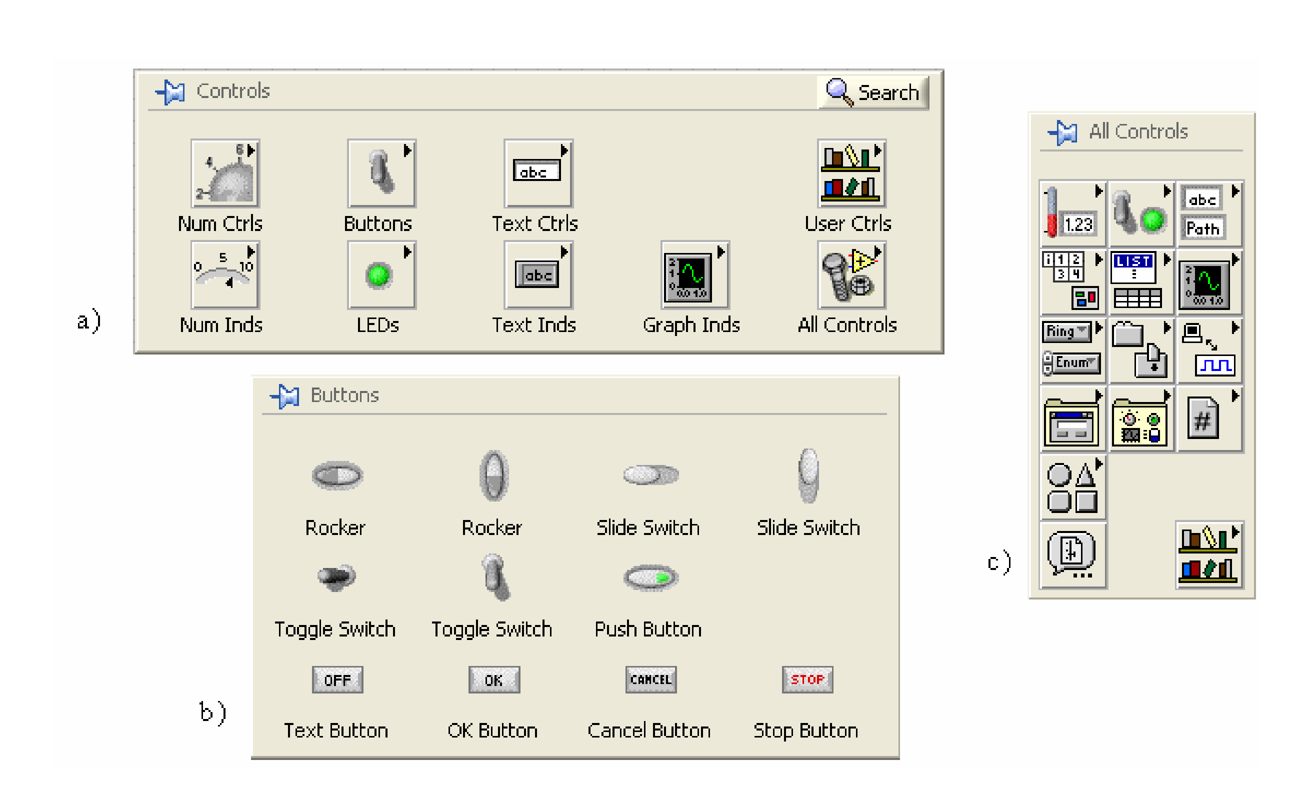


Abbildung A.4: a) die Elementepalette des Frontpanels
b) die Unterpalette „Buttons“
c) die vollständige Unterpalette „All Controls“

Funktionenpalette

Die Palette „Funktionen“ ist nur im Blockdiagramm verfügbar. Sie enthält die VIs und Funktionen zur Erstellung des Blockdiagramms. Diese sind ebenfalls je nach Typ in verschiedene Unterpaletten aufgeteilt. Es können verschiedene Ansichten gewählt werden, um unterschiedliche Bedien- und Anzeigeelemente darzustellen. Die Funktionenpalette kann durch Klicken auf „Fenster » Funktionenpalette“ oder mit der rechten Maustaste in den Arbeitsbereich des Blockdiagramms ausgewählt werden. Sie kann ebenfalls beliebig auf dem Bildschirm verschoben werden, die aktuelle Position der Palette wird auch hier beim Schließen von LabVIEW gespeichert.

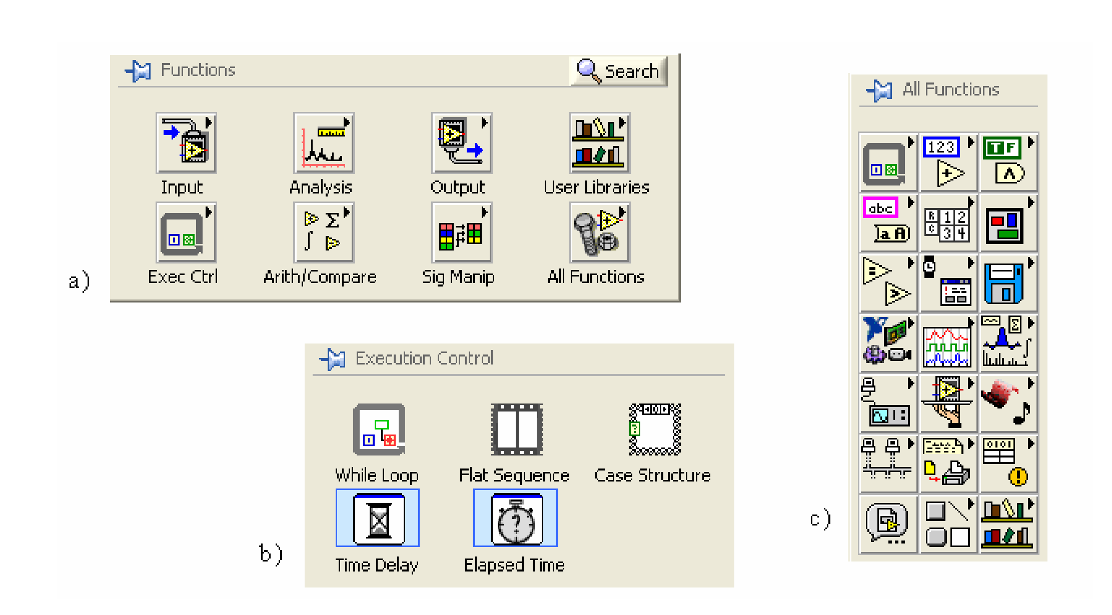


Abbildung A.5: a) die Funktionenpalette des Frontpanels
b) die Unterpalette „Execution Control“
c) die vollständige Unterpalette „All Functions“

Symbolleiste

Verwenden Sie die Schaltflächen der Symbolleiste, um ein VI auszuführen oder zu bearbeiten.

Kontexthilfe-Fenster

Wenn der Cursor über ein LabVIEW-Objekt bewegt wird, werden im Fenster der Kontexthilfe die wichtigsten Informationen hierzu angezeigt. Das gilt für VIs, Funktionen, Konstanten, Strukturen, Paletten, Eigenschaften, Methoden, Ereignissen und Komponenten von Dialogfeldern verfügbar. Die Kontexthilfe nützt auch beim Verdrahten von VIs oder Funktionen, eng beieinander liegende Anschlüsse voneinander zu unterscheiden.

Kapitel 3: Erstellen des Frontpanels

In der Regel sollten zuerst die Ein- und Ausgabeelementen auf dem Frontpanel positioniert werden, um ihnen dann im Blockdiagramm Funktionen zuzuweisen.

Konfiguration von Frontpanel-Objekten

Darstellungsart und Funktionsmerkmale von Frontpanel-Objekten können entweder über das Dialogfeld “Eigenschaften” oder über die Kontextmenüs des Frontpanels eingestellt werden. Das Dialogfenster “Eigenschaften” bietet die Möglichkeit, zu einem Bedien- oder Anzeigeelement mehrere Einstellungen auf einmal vorzunehmen. Ein weiterer Vorteil besteht darin, dass zu diesem Dialogfeld auch eine Kontext-Hilfe verfügbar ist. Kontextmenüs bieten sich dagegen zur schnellen Konfiguration einzelner Eigenschaften von Bedien- und Anzeigeelementen an. Grundsätzlich gilt, dass durch Änderungen, die über Kontextmenüs vorgenommen wurden, die entsprechenden Einstellungen im Dialogfeld überschrieben werden.

Durch Anklicken des Objekts auf dem Frontpanel mit der rechten Maustaste und der Wahl „Eigenschaften“ kann zu einem bestimmten Objekt das Dialogfenster “Eigenschaften” aufgerufen werden. Das gilt jedoch nur, wenn sich das VI im Bearbeitungsmodus befindet, während der Ausführung eines VIs sind diese Einstellungen nicht möglich.

Umwandlung von Bedien- in Anzeigeelemente und umgekehrt

Es gibt auch die Möglichkeit, ein Bedien- in ein Anzeigeelement umzuwandeln und umgekehrt. Dazu klickt man ein Objekt mit der rechten Maustaste an und wählt aus dem Kontextmenü die entsprechende Option.

Numerische Bedien- und Anzeigeelemente

Mit numerischen Bedien- und Anzeigeelementen lassen sich numerische Daten am einfachsten ein- und ausgeben. Die Größe kann horizontal geändert werden, um weitere Stellen anzuzeigen.

Der Wert eines numerischen Bedienelements oder einer numerischen Anzeige kann wie folgt geändert werden:

- Verwenden Sie das Bedienwerkzeug oder das Beschriftungswerkzeug, klicken Sie hiermit in das Fenster der numerischen Anzeige und geben Sie die Werte über die Tastatur ein.
- Platzieren Sie den Cursor mit Hilfe des Bedien- oder Beschriftungswerkzeugs rechts neben die zu ändernde Ziffer, und drücken Sie dann die Taste “Nach oben” oder “Nach unten”.

Festlegen des numerischen Formats

Per Voreinstellung erfolgt die Anzeige und Speicherung von Zahlen in LabVIEW wie in einem Taschenrechner. Das heißt, mit einem numerischen Bedien- bzw. Anzeigeelement können maximal sechs Stellen angezeigt werden. Wenn diese Anzahl überschritten wird, erfolgt eine automatische Umschaltung in die Exponentialdarstellung. Die Einstellung, ab wie vielen Stellen die Exponentialschreibweise verwendet werden soll, kann jedoch auch verändert werden. Dazu klickt man im Dialogfeld Eigenschaften numerischer Elemente auf die Registerkarte „Format und Genauigkeit“. Die ausgewählte Genauigkeit wirkt sich allerdings auf die Anzeige des Wertes aus. Die interne Genauigkeit richtet sich weiterhin nach dem Datentyp.

Tasten, Schalter und LEDs

Die booleschen Bedien- und Anzeigeelemente umfassen Nachbildungen von Tastern, Schaltern und LEDs. Sie dienen zur Eingabe bzw. Anzeige boolescher Werte (TRUE/FALSE). Wenn beispielsweise die Temperatur eines Experiments überwacht werden soll, kann man eine boolesche Warnleuchte auf das Frontpanel platzieren, um anzuzeigen, wann die Temperatur ein bestimmtes Niveau übersteigt. Über das Kontextmenü kann man das Erscheinungsbild eines booleschen Objekts den Bedürfnissen anpassen und festlegen, wie sich das Objekt verhalten soll, wenn es angeklickt wird.

Beschriftungen

Es empfiehlt sich, Beschriftungen zu verwenden, um Objekte auf dem Frontpanel und im Blockdiagramm zu kennzeichnen. In LabVIEW gibt es zwei Arten von Beschriftungen: mit Objekten verknüpfte und freie Beschriftungen. Verknüpfte Beschriftungen gehören zu einem speziellen Objekt, werden mit diesem verschoben und kennzeichnen nur dieses Objekt. Man kann eine verknüpfte Beschriftung zwar unabhängig verschieben, wenn man jedoch das mit der Beschriftung verknüpfte Objekt verschiebt, wird die Beschriftung zusammen mit dem Objekt verschoben. Verknüpfte Beschriftungen können ausgeblendet werden. Jedoch ist es nicht möglich, sie unabhängig vom zugehörigen Objekt zu kopieren oder zu löschen. Freie Beschriftungen sind nicht objektgebunden. Sie können unabhängig erstellt, verschoben, gedreht oder gelöscht werden. Freie Beschriftungen bieten sich daher zum Beispiel für Anmerkungen in Frontpanels und Blockdiagrammen an. Daneben eignen sie sich auch zur Dokumentation von Programmabschnitten im Blockdiagramm oder für Benutzeranweisungen auf dem Frontpanel.

Kapitel 4: Erstellen des Blockdiagramms

Nachdem das Frontpanel erstellt wurde, kann mit Hilfe grafisch dargestellter Funktionen Code hinzugefügt werden, um die Frontpanelobjekte zu steuern. Dieser grafische Quellcode ist im Blockdiagramm enthalten.

Datentypen für Bedien- und Anzeigeelemente

Tabelle A.6 enthält eine Auswahl von Symbolen für die unterschiedlichen Anschlusstypen bei Bedien- und Anzeigeelementen. Die Farbe und das Symbol des jeweiligen Anschlusses repräsentieren den Datentyp des Bedien- bzw. Anzeigeelements. Bedienelementanschlüsse weisen einen breiteren Rahmen als Anzeigeelementanschlüsse auf. Außerdem wird bei den Anschlüssen der Frontpanel-Elemente durch schwarze Pfeile angezeigt, ob es sich um ein Bedien- oder Anzeigeelement handelt. Befindet sich der Pfeil auf der rechten Seite des Anschlusses, handelt es sich um ein Bedien-, sonst um ein Anzeigeelement.

Bedien-element	Anzeige-element	Datentyp	Farbe	Standard werte
SGL	SGL	Fließkommazahl mit einfacher Genauigkeit	orange	0.0
DBL	DBL	Fließkommazahl mit doppelter Genauigkeit	orange	0.0
EXT	EXT	Fließkommazahl mit erweiterter Genauigkeit	orange	0.0
CSG	CSG	komplexe Fließkommazahl mit einfacher Genauigkeit	orange	0,0 + i0,0
CDB	CDB	komplexe Fließkommazahl mit doppelter Genauigkeit	orange	0,0 + i0,0
CXT	CXT	komplexe Fließkommazahl mit erweiterter Genauigkeit	orange	0,0 + i0,0
I8	I8	8-Bit-Ganzzahl mit Vorzeichen	blau	0
I16	I16	16-Bit-Ganzzahl mit Vorzeichen	blau	0
I32	I32	32-Bit-Ganzzahl mit Vorzeichen	blau	0
U8	U8	8-Bit-Ganzzahl ohne Vorzeichen	blau	0
U16	U16	16-Bit-Ganzzahl ohne Vorzeichen	blau	0
U32	U32	32-Bit-Ganzzahl ohne Vorzeichen	blau	0
TF	TF	boolesch	grün	FALSE
abc	abc	String	rosa	leerer String
[ ]	[ ]	Array—der Datentyp der Elemente wird in eckige Klammern eingeschlossen, und die Farbe des Datentyps wird übernommen.	unter-schiedlich	—

Abbildung A.6: Anschlüsse von Bedien- und Anzeigeelementen

Knoten

LabVIEW umfasst die folgenden Arten von Knoten:

- Funktionen: Integrierte Ausführungselemente, die mit einem Operator, einer Funktion oder einer Anweisung vergleichbar sind
- SubVIs: VIs, die in einem Blockdiagramm von einem anderen VI verwendet werden, vergleichbar mit Unterprogrammen
- Strukturen: Prozesssteuerungselemente wie beispielsweise Sequenzstrukturen, Case-Strukturen, For- oder While-Schleifen
- Formelknoten: In der Größe veränderbare Strukturen, mit denen Gleichungen direkt in ein Blockdiagramm eingegeben werden können

Überblick über Funktionen

Funktionen sind die grundlegenden Betriebselemente von LabVIEW. Die Elemente der Funktionen-Palette (mit einem hellgelben Hintergrund und einem schwarzen Vordergrund) sind die Symbole der Basisfunktionen. Funktionen verfügen nicht über Frontpanels oder Blockdiagramme, weisen jedoch Anschlussfelder auf. Sie können weder geöffnet noch bearbeitet werden. Die Palette „Funktionen“ beinhaltet auch die VIs, die zum Lieferumfang von LabVIEW gehören. Man kann diese VIs als SubVIs verwenden, wenn man VIs für die Datenerfassung, die Instrumentensteuerung, die Kommunikation oder andere VIs erstellt.

Numerische Funktionen

Mit den numerischen Funktionen können arithmetische, trigonometrische, logarithmische und komplexe mathematische Operationen durchgeführt und Zahlen in andere Datentypen konvertiert werden.

Boolesche Funktionen

Mit Hilfe der booleschen Funktionen können logische Operationen für einzelne boolesche Werte oder Arrays mit booleschen Werten durchgeführt werden:

- Ändern eines Wertes von TRUE zu FALSE und umgekehrt
- Umwandeln eines booleschen Wertes in eine Zahl (entweder 1 oder 0).

Array-Funktionen

Mit den Array-Funktionen können Arrays erstellt und verändert werden, zum Beispiel durch folgende Operationen:

- Extrahieren von einzelnen Datenelementen aus einem Array
- Hinzufügen von einzelnen Datenelementen zu einem Array
- Teilen von Arrays

Vergleichsfunktionen

Die Vergleichsfunktionen eignen sich sowohl zum Vergleichen von booleschen Werten als auch von Strings, numerischen Werten, Arrays und Clustern.

Zeit- und Dialogfunktionen

Mit den Zeit- und Dialogfunktionen sind folgende Operationen möglich:

- Manipulieren der Geschwindigkeit, mit der eine Operation ausgeführt wird
- Abrufen von Uhrzeit und Datum der systemeigenen Uhr
- Erstellen von Dialogfeldern für Anweisungen an den Benutzer

Datei-I/O-Funktionen

Mit den Datei-I/O-Funktionen sind folgende Operationen möglich:

- Öffnen und Schließen von Dateien
- Lesen aus und Schreiben in Dateien
- Schreiben von Strings, Zahlen, Arrays und Clustern in Dateien

Die Palette „Datei-I/O“ enthält auch VIs zum Ausführen von häufig anstehenden Datei-I/O-Aufgaben.

Signalverlaufsfunktionen

Mit den Signalverlaufsfunktionen sind folgende Operationen möglich:

- Erstellen von Signalverläufen, die Datenwerte sowie Informationen über Kanäle und Timing enthalten
- Extrahieren von einzelnen Datenelementen aus einem Signalverlauf
- Bearbeiten von einzelnen Datenelementen aus einem Signalverlauf

Hinzufügen von Anschlüssen zu Blockdiagrammfunktionen

Bei einigen Funktionen kann die Anzahl der Anschlüsse verändert werden. Wenn Sie beispielsweise ein Array mit zehn Elementen erstellen möchten, müssen Sie der Funktion “Array erstellen” zehn Anschlüsse hinzufügen. Um erweiterbaren VIs Anschlüsse hinzuzufügen, zieht man die Funktion mit dem Positionierwerkzeug nach oben oder unten. Mit dem Positionierwerkzeug lassen sich Anschlüsse auch wieder entfernen. Dazu dürfen sie jedoch nicht verbunden sein, andernfalls muss zuerst die vorhandene Verbindung zuerst gelöscht werden. Man kann Anschlüsse ebenso hinzufügen oder entfernen, indem man mit der rechten Maustaste auf einen der Anschlüsse der Funktion klickt und dann aus dem Kontextmenü „Eingang hinzufügen“, „Ausgang hinzufügen“, „Eingang entfernen“ oder „Ausgang entfernen“ wählt. Abhängig von der Funktion können dann Anschlüsse für Eingänge oder Ausgänge hinzugefügt werden. Mit dem Menüelement „Eingang hinzufügen“ und

„Ausgang hinzufügen“ wird ein Anschluss unmittelbar an den Anschluss angefügt, auf den man mit der rechten Maustaste geklickt hat. Wenn über das Kontextmenü ein bereits verbundener Anschluss entfernt wurde, wird der Anschluss gelöscht und die Verbindung getrennt.

Verbindung von Blockdiagrammobjekten

Die Datenübertragung zwischen den Blockdiagrammobjekten erfolgt über Verbindungsleitungen. Jede Verbindung besteht aus einer einzigen Datenquelle, die mit einer oder mehreren VIs oder Funktionen verbunden werden kann. Hierbei müssen alle notwendigen Anschlüsse verbunden werden. Ansonsten ist das VI nicht ausführbar. Die Anschlüsse, die verbunden werden müssen, um die Funktionstüchtigkeit eines Knotens zu gewährleisten, können über die Kontext-Hilfe eingesehen werden. Die entsprechenden Beschriftungen sind darin fett gedruckt. Je nach Datentyp haben die Verbindungen unterschiedliche Farben, Formate und Linienstärken. Verbindungsstümpfe sind die abgeschnittenen Leitungen, die an einem unverbundenen VI- oder Funktionssymbol angezeigt werden, wenn man das Verbindungswerkzeug über das Symbol bewegt. Sie zeigen den Datentyp des jeweiligen Anschlusses an. Darüber hinaus erscheint auch ein Hinweisstreifen mit dem Namen des Anschlusses. Ist der Anschluss bereits verbunden, erscheint der Verbindungsstumpf für diesen Anschluss nicht mehr, wenn man das Verbindungswerkzeug über die Funktion bewegt. Ein Verbindungssegment ist ein einziges, horizontal oder vertikal verlegtes Verbindungsstück. An der Stelle, an der zwei Segmente verbunden werden, entsteht eine Abzweigung. Der Punkt, an dem zwei oder mehr Verbindungssegmente zusammenlaufen, wird als Knotenpunkt bezeichnet. Ein Verbindungszweig enthält alle Verbindungssegmente von Knotenpunkt zu Knotenpunkt, von Anschluss zu Knotenpunkt oder von Anschluss zu Anschluss, wenn sich keine Knotenpunkte dazwischen befinden.

Manuelles Verbinden von Objekten

Man verwendet das Verbindungswerkzeug, um Anschlüsse an einem Blockdiagrammknoten mit den Anschlüssen an einem anderen Blockdiagrammknoten zu verbinden. Die Cursorspitze des Werkzeugs ist die Spitze der abgewickelten Leitungsspule. Wenn Sie das Verbindungswerkzeug über einen Anschluss bewegen, blinkt dieser. Bei Anschlüssen von VIs oder Funktionen erscheint daneben auch ein Hinweisstreifen mit dem Namen des Anschlusses. Wenn man zwei inkompatible Anschlüsse miteinander verbinden will, erscheint am Verbindungswerkzeug ein Warnzeichen. Die fehlerhafte Verbindung kann zwar dennoch erstellt, muss jedoch vor Ausführung des VIs entfernt werden, damit es funktionstüchtig ist.

Erstellen von Verbindungen

Beim Verdrahten sucht LabVIEW automatisch nach einem günstigen Verbindungsweg. So werden Verbindungen zum Beispiel um vorhandene Objekte wie Schleifen oder Strukturen herumgelegt. Außerdem werden alle Verbindungen soweit wie möglich begradigt, also optimiert. Dabei wird nach Möglichkeit so verfahren, dass alle Verbindungen aus Bedienelementen an der rechten Seite des Symbols austreten und bei Anzeigeelementen an der linken Seite eintreten.

Markieren von Verbindungen

Zum Markieren von Verbindungen klickt man diese mit dem Positionierwerkzeug entweder ein-, zwei- oder dreimal an. Mit einem einfachen Klick auf eine Verbindung wird ein Segment der Verbindung markiert. Mit einem Doppelklick wird ein Verbindungszweig markiert. Wenn man dreimal auf eine Verbindung klickt, wird die ganze Verbindung markiert.

Beseitigen von Verbindungsfehlern

Für unterbrochene Verbindungen gibt es eine Vielzahl von Gründen, beispielsweise wenn man versucht, zwei Objekte mit inkompatiblen Datentypen miteinander zu verbinden. Wenn man das Verbindungswerkzeug über eine unterbrochene Verbindung bewegt, wird ein Hinweisstreifen angezeigt, in dem erläutert wird, warum die Verbindung fehlerhaft ist. Gleichzeitig erscheint diese Information auch im Fenster „Kontext-Hilfe“. Klickt man mit der rechten Maustaste auf die Verbindung, und wählt im Kontextmenü „Fehler auflisten“ aus, wird das Fenster „Fehlerliste“ angezeigt. Weitere Informationen darüber, warum die jeweilige Verbindung unterbrochen ist, erhält man durch einen Klick auf die Schaltfläche LabVIEW-Hilfe. Klickt man mit dem Positionierwerkzeug auf die Verbindung, kann mit der <Entf>-Taste eine unterbrochene Verbindung gelöscht werden. Um alle fehlerhaften Verbindungen zu löschen, wählt man „Bearbeiten » Ungültige Verbindungen entfernen“ oder drückt die Tastenkombination <Strg-B>.

Typumwandlungspunkte

Wenn zwei unterschiedliche Datentypen miteinander verbunden werden, werden auf dem Blockdiagrammknoten Typumwandlungspunkte angezeigt, mit denen auf die unterschiedliche Darstellung hingewiesen werden soll. Der Punkt bedeutet, dass der an den Knoten weitergeleitete Wert von LabVIEW in eine andere Darstellung konvertiert wurde. So sind zum Beispiel für die Additionsfunktion Gleitkommazahlen mit doppelter Genauigkeit als Eingangswerte zulässig. Wenn man einen der Eingänge für ganzzahlige Werte konfiguriert, erscheint an der Funktion ein Typumwandlungspunkt.

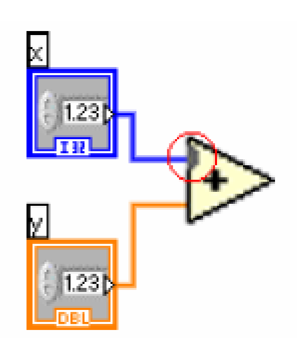


Abbildung A.7: Typumwandlungspunkt

Im Blockdiagramm wird am Rand des Anschlusses ein Typumwandlungspunkt angezeigt, um darauf hinzuweisen, dass eine automatische numerische Umwandlung erfolgt ist. Da VIs und Funktionen über viele Anschlüsse verfügen können, kann ein Typumwandlungspunkt auch im Inneren des Symbols angezeigt werden, wenn man einen Anschluss mit einem anderen verbindet. Wenn ein VI Typumwandlungspunkte enthält, kann es langsamer und speicheraufwändiger werden. Daher sollten in VIs nach Möglichkeit immer einheitliche Datentypen verwendet werden.

Datenflussprinzip Blockdiagramm

Für das Ausführen von VIs folgt LabVIEW einem Datenflussmodell. Ein Blockdiagrammknoten wird ausgeführt, wenn an allen seinen Eingängen Daten verfügbar sind. Wenn die Ausführung eines Knotens abgeschlossen ist, werden die Daten an die Ausgangsanschlüsse übergeben und diese Ausgabedaten an den nächsten Knoten im Datenflusspfad weitergeleitet. Visual Basic, C++, JAVA und die meisten anderen textbasierten Programmiersprachen folgen bei der Programmausführung einem Steuerflussmodell. Hier bestimmt die sequenzielle Reihenfolge der Programmelemente die Ausführungsreihenfolge eines Programms. Da in LabVIEW der Datenfluss und nicht die sequenzielle Reihenfolge der Befehle die Ausführungsreihenfolge der Blockdiagrammelemente bestimmt, kann man Blockdiagramme erstellen, die simultane Operationen beinhalten. So können zum Beispiel zwei While-Schleifen gleichzeitig ausgeführt und die Ergebnisse auf dem Frontpanel angezeigt werden.

Entwerfen des Blockdiagramms

Bei der Erstellung von Blockdiagrammen sollten folgende Hinweise beachtet werden:

- Gehen Sie beim Aufbau des Blockdiagramms immer von links nach rechts und von oben nach unten vor. Obwohl die Position der Blockdiagrammelemente keinen Einfluss auf die Ausführungsreihenfolge hat, wirkt das Blockdiagramm geordneter und ist einfacher zu verstehen, wenn Sie eine Datenflussrichtung

beibehalten. Die Ausführungsreihenfolge wird ausschließlich durch die Verbindungen und Strukturen bestimmt.

- Vermeiden Sie es, Blockdiagramme zu erstellen, die mehr als einen oder zwei Bildschirmbreiten in Anspruch nehmen. Wenn ein Blockdiagramm umfangreich und komplex wird, ist es schwieriger zu verstehen und die Fehlersuche wird unter Umständen erschwert.
- Prüfen Sie, ob Sie einige Komponenten des Blockdiagramms in anderen VIs wiederverwenden können oder ob ein Abschnitt des Blockdiagramms als logische Komponente zusammengefasst werden kann. Falls ja, teilen Sie das Blockdiagramm in SubVIs auf, die bestimmte Aufgaben erledigen. Die Verwendung von SubVIs hilft bei der Änderungsverwaltung und sorgt dafür, dass Fehler in Blockdiagrammen schnell behoben werden können.
- Gestalten Sie das Blockdiagramm übersichtlich, indem Sie für kurze und klare Verbindungen sorgen. Blockdiagramme sind zwar auch funktionstüchtig, wenn sie ineffizient verdrahtet sind, können jedoch nur schwierig zu lesen sein und im Fehlerfall die Suche erschweren. Auch kann in einem unübersichtlichen Blockdiagramm der Anschein entstehen, dass das VI Funktionen ausführt, die nicht vorgesehen sind.
- Vermeiden Sie es, Verbindungen unter einen Strukturrahmen oder zwischen sich überlappenden Objekten zu ziehen, da LabVIEW in diesem Fall möglicherweise einige Verbindungssegmente überdeckt.
- Vermeiden Sie es, Objekte über Verbindungen zu platzieren. Mit Verbindungen werden nur die Objekte miteinander verknüpft, auf die Sie klicken. Wenn Sie einen Anschluss oder ein Symbol auf eine Verbindung setzen, könnte der Eindruck entstehen, es bestünde eine Verbindung.

Kapitel 5: Ausführen von VIs und Fehlersuche

Zum Ausführen eines VIs müssen zunächst alle SubVIs, Funktionen und Strukturen mit den für die Anschlüsse zulässigen Datentypen verbunden werden. Es kann vorkommen, dass ein VI unerwartete Daten produziert oder nicht wie geplant abläuft. Mit Hilfe von LabVIEW kann man die Ausführung von VIs konfigurieren und Fehler in der Blockdiagrammanordnung bzw. im Programmablauf erkennen. Abbildung A.8 zeigt die wichtigsten Symbole der Kopfzeile.

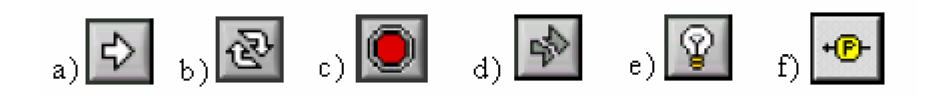


Abbildung A.8: Wichtige Buttons der Symbolleiste

Ausführen von VIs

Beim Ausführen eines VIs wird die Funktion realisiert, für die das VI erstellt worden ist. Ein VI ist immer dann ausführbar, wenn die Schaltfläche „Ausführen“ auf der Symbolleiste als weißer Pfeil (Abb.8, links) erscheint. Außerdem zeigt der weiße Pfeil an, dass das VI als SubVI verwendet werden kann, wenn für das VI ein Anschlussfeld erstellt wurde. Bei Betätigung der Schaltfläche Ausführen wird der Pfeil wie in der Abbildung links schwarz dargestellt. Solange das VI läuft, sind keine Änderungen am Blockdiagramm möglich.

Das VI wird ausgeführt, wenn man entweder auf die Schaltfläche „Ausführen“ (Abbildung A.8a), „wiederholt ausführen“ (Abbildung A.8b)) oder die Einzelschrittschaltflächen auf der Blockdiagramm-Symbolleiste klickt. Mit dem Klicken auf die Schaltfläche „Ausführen“ wird das VI einmal ausgeführt. Das VI stoppt, wenn der Datenfluss abgeschlossen ist. Bei Betätigung der Schaltfläche „wiederholt ausführenW läuft ein VI so lange kontinuierlich ab, bis es per Hand wieder angehalten wird. Wenn man auf die Einzelschrittschaltflächen klickt, wird das VI schrittweise ausgeführt.

Fehlerbeseitigung in nicht ausführbaren VIs

Wenn ein VI nicht gestartet werden kann, enthält es einen Fehler. Die Schaltfläche „Ausführen“ ist dann gebrochen dargestellt (Abbildung A.8d). Häufig erscheint der Pfeil auch während der Erstellung und Bearbeitung eines VIs gebrochen, solange dieses noch fehlerhaft ist. Wenn er sich nach Fertigstellung aller Verbindungen im Blockdiagramm nicht in einen intakten Pfeil umwandelt, ist das VI nicht ausführbar.

Fehlersuche in nicht ausführbaren VIs

Man kann auf die gebrochene Schaltfläche „Ausführen“ klicken oder „Fenster » Fehlerliste anzeigen“ wählen, um herauszufinden, warum ein VI nicht ausgeführt werden kann. Im Fenster „Fehlerliste“ werden alle Fehler aufgeführt. Im Abschnitt „VI-Liste“ werden die Namen aller VIs im Speicher aufgeführt, die fehlerhaft sind. Im Abschnitt „Fehler und Warnungen“ werden die Fehler und Warnungen für dasjenige VI angezeigt, das Sie im Abschnitt VI-Liste markiert haben. Im Abschnitt „Details“ werden die Fehler und in manchen Fällen auch Empfehlungen zur Fehlerbehebung und Hinweise auf weitere Informationen hierzu angezeigt. Klickt man auf die Schaltfläche „Fehler anzeigen“ oder doppelt auf die Fehlerbeschreibung, so wird das Blockdiagramm oder Frontpanel angezeigt und das Objekt hervorgehoben, das den Fehler enthält.

Highlight-Funktion

Über die Schaltfläche für die Highlight-Funktion (Abbildung A.8e) kann im Blockdiagramm eine Ausführungsanimation angezeigt werden. Durch die Highlight-Funktion wird mit Hilfe von Kreisen der Datenfluss von einem Knoten zum anderen im Blockdiagramm veranschaulicht. Verwendet man die Highlight-Funktion in Verbindung mit der Einzelschrittausführung, so sieht man, wie sich die Daten von Knoten zu Knoten durch ein VI bewegen. Bei Verwendung der Highlight-Funktion wird die Ausführungsgeschwindigkeit des VIs erheblich herabgesetzt.

Sonden-Werkzeug (Probes)

Mit dem Sonden-Werkzeug (siehe Abbildung A.8 f)) können während der Ausführung eines VIs Zwischenwerte an Verbindungsstücken angezeigt werden. Das Sonden-Werkzeug eignet sich besonders für komplizierte Blockdiagramme mit einer Reihe von Operationen, von denen jede inkorrekte Daten zurückgeben könnte. Verwendet man das Sonden-Werkzeug zusammen mit der Highlight-Funktion, lässt sich feststellen, ob und wo falsche Werte auftreten.

Verwendung von SubVIs

LabVIEW enthält VIs und Funktionen, mit denen spezifische Applikationen erstellt werden können, wie beispielsweise Datenerfassungs-VIs und -Funktionen oder VIs, die auf andere VIs zugreifen bzw. mit anderen Applikationen kommunizieren. Diese VIs können als SubVIs in Ihrer Applikation verwendet werden, um Entwicklungszeit einzusparen. Nachdem man ein VI erstellt und dessen Symbol und Anschlussfeld erzeugt hat, kann man es in einem anderen VI verwenden. Ein VI, das im Blockdiagramm eines anderen VIs aufgerufen wird, nennt man SubVI. Ein SubVI

entspricht in textbasierten Programmiersprachen einem Unterprogramm. Ein SubVI-Knoten entspricht einem Subroutinenaufruf in textbasierten Programmiersprachen. Der SubVI-Knoten ist nicht das eigentliche SubVI, ebenso wenig wie die Aufrufanweisung für eine Subroutine in einem Programm mit der Subroutine selbst zu verwechseln ist. Ein Blockdiagramm mit mehreren identischen SubVI-Knoten ruft dasselbe SubVI mehrere Male auf. Die Bedien- und Anzeigeelemente eines SubVIs empfangen Daten vom und geben Daten an das Blockdiagramm des aufrufenden VIs zurück. Klickt man in der Palette „Funktionen“ auf das Symbol „VI auswählen“, kann ein VI ausgewählt werden und in das Blockdiagramm platziert werden, damit es als SubVI aufgerufen wird.

Zum Erstellen des Messprogramms zum Versuch „Lange Halbwertszeiten“ stehen vier SubVIs zur Verfügung, die die Ansteuerung der Geräte beinhalten.

Kapitel 6 – Schleifen und Strukturen

Schleifen und Strukturen

Strukturen sind grafische Darstellungen der Schleifen und Case-Anweisungen in textbasierten Programmiersprachen. Man verwendet Strukturen im Blockdiagramm, um Codeblöcke zu wiederholen und den Code bedingungsabhängig oder in einer bestimmten Reihenfolge auszuführen. Wie andere Knoten verfügen auch Strukturen über Anschlüsse, über die sie mit anderen Blockdiagrammknoten verbunden werden können. Sie werden automatisch ausgeführt, wenn Eingabedaten verfügbar sind, und liefern Daten an Ausgabeverbindungen, wenn die Ausführung abgeschlossen ist. Jede Struktur hat einen markanten, in der Größe veränderbaren Rahmen, mit dem der Abschnitt des Blockdiagramms umschlossen wird, der entsprechend den Regeln der Struktur ausgeführt wird. Ein Blockdiagrammabschnitt innerhalb einer Struktur wird als Subdiagramm bezeichnet. Die Anschlüsse, die Daten an Strukturen übergeben, bzw. Daten aus Strukturen übernehmen, werden Tunnel genannt. Ein Tunnel ist ein Verbindungspunkt an einem Strukturrahmen.

Mit folgenden Strukturen, die sich auf der Palette Strukturen befinden, kann gesteuert werden, wie in einem Blockdiagramm Prozesse ausgeführt werden:

- FOR-Schleife: wiederholt die Ausführung eines Subdiagramms so oft wie vorgegeben
- While-Schleife: führt ein Subdiagramm so lange aus, bis eine bestimmte Bedingung erfüllt ist
- Case-Struktur: enthält mehrere Rahmen, von denen je nach dem Wert, der an der Struktur anliegt, jeweils einer ausgeführt wird
- Sequenzstruktur: enthält ein oder mehrere Subdiagramme, die nacheinander ausgeführt werden
- Formelknoten: führt mathematische Operationen auf Grundlage einer numerischen Eingabe durch
- Ereignisstruktur: enthält einen oder mehrere Rahmen, die dann ausgeführt werden, wenn während des Programmablaufs vom Benutzer bestimmte Ereignisse ausgelöst werden

FOR- und While-Schleifenstrukturen

Mit FOR- und While-Schleifen ist es möglich, sich wiederholende Operationen zu steuern.

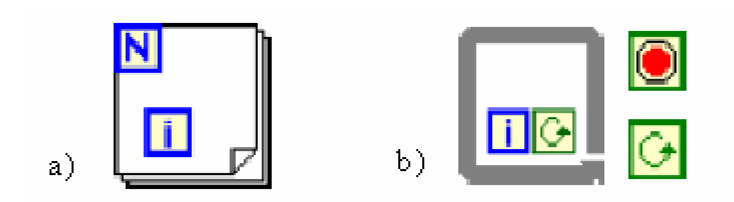


Abbildung A.9: a) For-Schleife und b) While-Schleife

FOR-Schleifen (Abb. A.9a)

Durch den Eingabewert des Schleifenzählers N wird bestimmt, wie oft das Subdiagramm wiederholt werden soll. Sie können die Anzahl explizit einstellen, indem Sie einen Wert von außerhalb der Schleife mit der linken oder der oberen Seite des Zähl-Anschlusses verbinden, oder Sie können die Anzahl implizit mit der Auto-Indizierung festlegen. Der Iterationsanschluss gibt die Anzahl der abgeschlossenen Schleifendurchläufe aus. Die Zählung beginnt dabei immer bei Null. Das heißt, während des ersten Durchlaufs gibt der Iterationsanschluss den Wert 0 aus. Sowohl der Schleifenzähler als auch der Iterationsanschluss sind vom Typ "vorzeichenbehafteter Long Integer". Wenn Sie eine Fließkommazahl mit dem Zähl-Anschluss verbinden, rundet LabVIEW diese Zahl und zwingt sie innerhalb des Bereichs. Wenn am Schleifenzähler der Wert 0 oder ein negativer Wert anliegt, wird die Schleife nicht ausgeführt und an den Ausgängen werden die für den jeweiligen Datentyp voreingestellten Werte ausgegeben.

Es lassen sich der FOR-Schleife Schieberegister hinzufügen, um Daten aus der aktuellen Wiederholung an die nächste Wiederholung zu übergeben.

While-Schleifen

Ähnlich einer „Do-Schleife" oder einer „Repeat-Until-Schleife" in textbasierten Programmiersprachen wird bei einer While-Schleife (Abbildung A.9 b)) ein Subdiagramm so lange ausgeführt, bis eine bestimmte Bedingung zutrifft. Die While-Schleife führt das Subdiagramm aus, bis der Bedingungsanschluss, ein Eingabeanschluss, einen bestimmten booleschen Wert empfängt. Der Bedingungsanschluss ist standardmäßig auf „Weiter wenn TRUE" (Abbildung A.9 b) rechts oben) eingestellt. Wenn der Bedingungsanschluss auf „Stopp wenn TRUE" eingestellt ist, wird die Schleife so lange ausgeführt, bis der Wert TRUE anliegt. Um den Anschluss auf „Weiter wenn TRUE" zu stellen, klickt man mit der rechten Maustaste auf den Rahmen der Schleife und wählt die entsprechende Option aus dem

Kontextmenü. In diesem Fall wird die Schleife so lange ausgeführt, bis der Wert FALSE anliegt. Man kann den Bedingungsanschluss auch mit dem Bedienwerkzeug anklicken, um die Bedingung zu ändern. Der Iterationsanschluss gibt die Anzahl der abgeschlossenen Schleifendurchläufe aus. Die Zählung beginnt wie bei der For-Schleife immer bei Null. Das heißt, während des ersten Durchlaufs gibt der Iterationsanschluss den Wert 0 aus. Auch der While-Schleife lassen sich Schieberegister hinzufügen, um Daten aus der aktuellen Wiederholung an die nächste Wiederholung zu übergeben.

Schieberegister

Mit Schieberegistern und Rückkopplungsknoten werden in FOR- und While-Schleifen Werte von einem Schleifendurchlauf zum nächsten übergeben.

Ein Schieberegister wird als ein Paar von Anschlüssen dargestellt, die sich an den Längsseiten der Schleife gegenüber liegen. Der rechte Anschluss (durch einen Pfeil nach oben gekennzeichnet) speichert die Werte eines jeden Durchlaufs. Diese werden jeweils in den nächsten Durchlauf übernommen. Zur Erstellung eines Schieberegisters klickt man mit der rechten Maustaste auf den linken oder rechten Rand einer Schleife und wählt im Kontextmenü die Option „Schieberegister hinzufügen“. Mit Schieberegistern kann jeder beliebige Datentyp übertragen werden, da sie sich automatisch auf den Datentyp des ersten Objekts einstellen, das mit dem Schieberegister verbunden ist. Die Daten, die an die Anschlüsse des jeweiligen Schieberegisters übergeben werden, müssen vom gleichen Typ sein. Zur Initialisierung eines Schieberegisters verbindet man den Anschluss auf der linken Seite der Schleife mit einem Bedienelement oder einer Konstante. Damit wird der Wert zurückgesetzt, den das Schieberegister beim Start der Schleifenausführung weitergibt. Wenn das Register nicht initialisiert wird, verwendet die Schleife den Wert der letzten Schleifenausführung oder den Standardwert für den entsprechenden Datentyp, wenn die Schleife noch nicht ausgeführt wurde. Wenn also bei jedem VI-Start der zuletzt ausgegebene Wert als Anfangswert verwendet werden soll, muss das Register nicht initialisiert werden. Auf diese Weise können zum Beispiel Statusinformationen für die nachfolgende Ausführung des VIs gespeichert werden. Nachdem die Schleife ausgeführt wurde, verbleibt der letzte im Schieberegister gespeicherte Wert im rechten Anschluss. Eine Schleife kann auch mehrere Schieberegister enthalten. Dadurch können die Werte mehrerer verschiedener Operationen in die jeweils nächste Schleifenausführung übernommen werden.

Case- und Sequenzstrukturen

In Case-, Sequenz- und Ereignisstrukturen sind immer mehrere Subdiagramme enthalten. Eine Case-Struktur führt ein Subdiagramm in Abhängigkeit von dem an die Struktur übergebenen Eingabewert aus. Bei der flachen und der gestapelten Sequenzstruktur werden die darin enthaltenen Subdiagramme nacheinander ausgeführt. In einer Ereignis-Struktur hängt die Ausführung des Subdiagramms davon ab, wie der Benutzer auf das VI Einfluss nimmt.

Case-Strukturen

Eine Case-Struktur (Abbildung A.10) hat immer mindestens zwei Subdiagramme oder Cases. Davon ist immer nur jeweils ein Subdiagramm sichtbar, und die Struktur führt immer nur jeweils einen Case aus. Welches Subdiagramm ausgeführt wird, hängt vom jeweiligen Eingabewert ab. Die Case-Struktur entspricht der CASE- bzw. if...then...else-Anweisung in textbasierten Programmiersprachen. Im Case-Selektor am oberen Rand der Struktur wird der Name des jeweiligen Rahmens sowie die Dekrement- und Inkrementschaltfläche auf jeder Seite angezeigt. Damit kann zwischen den einzelnen Rahmen umgeschaltet werden. Wenn man auf den Pfeil nach unten klickt, der sich rechts neben der Case-Beschriftung befindet, wird ein Pulldown-Menü mit den vorhandenen Cases angezeigt.

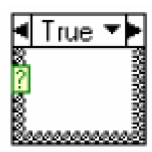


Abbildung A.10: Case-Struktur

Um festzulegen, welcher Case ausgeführt werden soll, verbindet man einen Eingabewert mit dem Selektoranschluss, der als ein Fragezeichen auf dem Rand der Struktur dargestellt ist (Abb. A.10). Dabei kann es sich um einen Integer-, einen booleschen Wert, einen String oder einen Wert vom Enum-Typ handeln. Der Selektoranschluss kann an jede beliebige Stelle am linken Rand der Case-Struktur platziert werden. Wenn ein boolescher Wert an die Struktur übergeben wird, enthält diese einen TRUE- und einen FALSE-Case. Bei Integer-Werten und Enum-Werten oder Strings kann die Struktur beliebig viele Cases haben.

Für den Fall, dass Werte außerhalb des zulässigen Bereichs auftreten, sollte bei Case-Strukturen immer ein Standard-Case festgelegt werden. Ansonsten muss für jeden möglichen Eingabewert ein Case erstellt werden. Wenn der Selektoranschluss zum Beispiel vom Typ Integer ist, und Cases für die Eingangswerte 1, 2 und 3 eingerichtet

sind, sollte auch ein Standard-Case für den Fall vorhanden sein, dass der Eingabewert unerwartet ungleich 1, 2 oder 3 ist.

Eingabe- und Ausgabetunnel

Eine Case-Struktur kann mehrere Ein- und Ausgabetunnel enthalten. Die Eingänge stehen allen Rahmen zur Verfügung, müssen jedoch nicht für jeden Rahmen verwendet werden. Die Ausgabetunnel müssen dagegen für jeden Case festgelegt werden. Wenn für einen Case ein Ausgabetunnel erstellt wird, erscheinen bei allen anderen Cases an derselben Position am Rahmen ebenfalls Tunnel. Wenn ein Case keinen Wert an den Tunnel übergibt, erscheinen alle Tunnel als weiße Quadrate. Die Datenquelle eines Ausgabetunnels kann für jeden Case unterschiedlich sein. Die verwendeten Datentypen müssen jedoch zueinander kompatibel sein. Wenn man den Ausgabetunnel mit der rechten Maustaste anklickt und aus dem Kontextmenü die Option „Standardwert für offene Verbindung“ auswählt, werden für alle nicht verbundenen Tunnel Standardwerte verwendet.

Sequenzstrukturen

Sequenzstrukturen sind dadurch gekennzeichnet, dass die darin enthaltenen Rahmen in sequentieller Reihenfolge ausgeführt werden. Sie können jedoch auch nur aus einem Subdiagramm bestehen. Es gibt zwei Arten von Sequenzstrukturen: flache und gestapelte (Abbildung A.11).

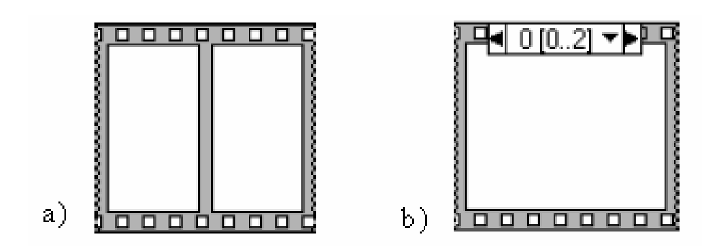


Abbildung A.11: a) flache Sequenzstruktur und b) gestapelte Sequenzstruktur

Flache Sequenzstruktur

Bei einer flachen Sequenzstruktur (Abbildung A.11a) werden alle Rahmen nebeneinander angezeigt und von links nach rechts ausgeführt. Mit flachen Sequenzstrukturen lässt sich eine zu häufige Verwendung von lokalen Sequenzvariablen vermeiden und die Dokumentation des Blockdiagramms verbessern. Beim Hinzufügen oder Löschen von Rahmen passt sich die Größe der Struktur automatisch an. Rahmen können auch über Ausschneiden und Einfügen neu angeordnet werden.

Gestapelte Sequenzstruktur

Bei einer gestapelten Sequenzstruktur (Abbildung A.11b) werden die Rahmen übereinander dargestellt, so dass immer jeweils ein Rahmen sichtbar ist. Die Abarbeitung erfolgt der Reihe nach, bei Rahmen 0 beginnend. Die Daten werden immer erst nach Beendigung des letzten Rahmens ausgegeben. Die gestapelte Sequenzstruktur empfiehlt sich insbesondere um Platz auf dem Blockdiagramm zu sparen. In der Beschriftung am oberen Rand der gestapelten Sequenzstruktur wird die Nummer des aktuellen Rahmens und die Gesamtzahl der Rahmen angezeigt. Hier können die verfügbaren Rahmen durchgeblättert und neu angeordnet werden. Die Rahmenbeschriftung am oberen Rand der Sequenzstruktur ähnelt der Selektorkennung in einer Case-Struktur. Sie enthält in der Mitte die Rahmennummer und an den Außenkanten Pfeile zur Rahmenauswahl. Man klickt auf die Pfeile, um sich die verfügbaren Rahmen anzeigen zu lassen. Bei einem Klick auf den nach unten zeigenden Pfeil rechts neben der Rahmennummer wird ein Kontextmenü mit den verfügbaren Rahmen geöffnet. Zum Umsortieren der Rahmen klickt man mit der rechten Maustaste auf den Rand des Rahmens und wählt aus dem Kontextmenü die Option „Diesen Rahmen zu ... setzen“ mit der entsprechenden Rahmennummer aus. Im Gegensatz zu Case-Strukturen können bei Sequenzstrukturen keine Werte in die Beschriftung eingegeben werden. Wenn Rahmen hinzugefügt, entfernt oder in ihrer Anordnung verändert werden, wird die Nummerierung automatisch angepasst.

Einsatz von Sequenzstrukturen

Sequenzstrukturen sollten verwendet werden, um für Blockdiagrammabschnitte eine Reihenfolge in der Ausführung festzulegen, bei denen keine natürliche Datenabhängigkeit vorliegt. Datenabhängigkeit bedeutet, dass ein Knoten B erst ausgeführt werden kann, wenn der Knoten A, von dem dieser Daten erhält, die Ausführung beendet hat. Die Ausführungsreihenfolge wird innerhalb der einzelnen Rahmen einer Sequenzstruktur wie im Rest des Blockdiagramms durch die Datenabhängigkeit bestimmt. Anders als bei Case-Strukturen kann es bei gestapelten Sequenzstrukturen zu jedem Tunnel immer nur eine Datenquelle geben. Zwar kann die Ausgabe grundsätzlich von jedem Rahmen erfolgen, jedoch muss die Ausführung aller Rahmen immer abgeschlossen sein. Wie bei Case-Strukturen stehen die Daten an den Eingabetunneln für alle Rahmen zur Verfügung.

Kapitel 7 - Arrays

Die Array-Bedienelemente und Array-Funktionen dienen zur Gruppierung von Daten. Mit Arrays werden Datenelemente gleichen Typs zusammengefasst.

Arrays

Ein Array besteht aus Elementen und Dimensionen. Die Elemente sind die Daten, aus denen sich das Array zusammensetzt. Eine Dimension ist die Länge, Höhe oder Tiefe eines Arrays. Ein Array kann eine oder mehrere Dimensionen und je nach verfügbarem Speicher bis zu $2^{31}-1$ Elemente pro Dimension enthalten. Arrays können aus numerischen, booleschen, Pfad-, String-, Signalverlaufs- und Cluster-Daten erstellt werden. Der Einsatz von Arrays bietet sich insbesondere an, wenn ähnliche Daten verwendet oder Berechnungen wiederholt werden sollen. Arrays eignen sich ideal für das Speichern von Signalverlaufsdaten oder von Daten, die in Schleifen erzeugt werden. In diesem Fall wird bei jedem Schleifendurchlauf ein neues Array-Element erzeugt.

Indizes

Um ein bestimmtes Element in einem Array zu finden, muss ein Index pro Dimension vorhanden sein. In LabVIEW können mit Hilfe von Indizes Elemente, Zeilen, Spalten und Seiten aus einen Array im Blockdiagramm abgerufen werden.

Beispiele für Arrays

Ein Beispiel für ein numerisches Array sind die Abtastwerte eines Signalverlaufs – jede Zelle enthält einen Abtastwert bezogen auf jeweils einen Abtastzeitpunkt (Abbildung A.12):

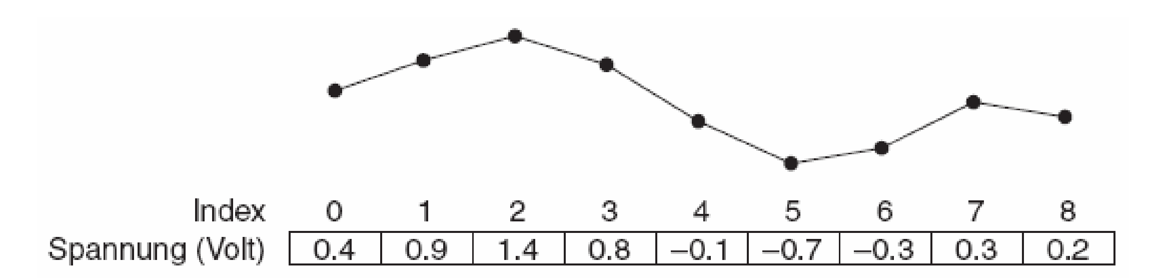


Abbildung A.12: Spannungsverlauf als ein aus Zahlen bestehendes Array

In dem vorstehenden Beispiel wird ein 1D-Array verwendet. Bei einem 2D-Array werden die Elemente in einem Raster gespeichert. Dies setzt einen Spalten- und einen Zeilenindex zum Suchen eines Elements voraus, die beide jeweils nullbasiert sind, was bedeutet, dass die erste Spalte 0, die zweite Spalte 1 usw. ist.

In Abbildung A.13 ist also ein aus 6 Spalten und 4 Zeilen bestehendes 2D-Array dargestellt, das 24 Elemente enthält.

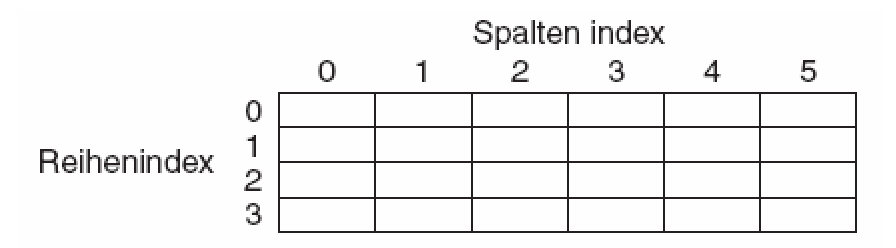


Abbildung A.13: Spannungsverlauf als ein aus Zahlen bestehender Array

Beschränkungen für Arrays

Man kann aus beinahe jedem Datentyp ein Array erstellen, wobei die folgenden Ausnahmen gelten. Es ist nicht möglich

- Arrays aus Unterpanel-Bedienelementen zu erzeugen
- Arrays aus Registerbedienelementen zu erzeugen
- Arrays aus ActiveX-Bedienelementen zu erzeugen
- Arrays aus Diagrammen zu erzeugen
- Arrays aus XY-Graphen mit mehreren Kurven zu erzeugen

Array-Funktionen

Mit den Array-Funktionen können Arrays erstellt und verändert werden, z.B. durch folgende Operationen:

- Extrahieren von einzelnen Datenelementen aus einem Array
- Einfügen, Löschen oder Ersetzen von Datenelementen in einem Array
- Teilen von Arrays

Automatisches Ändern der Größe von Array-Funktionen

Die Funktionen „Array indizieren”, „Teilarray ersetzen”, „In Array einfügen”, „Aus Array entfernen” und „Teilarray” passen sich automatisch an die Anzahl an Dimensionen des angeschlossenen Eingabe-Arrays an. Wenn man beispielsweise ein 1D-Array mit einer dieser Funktionen verbindet, hat die Funktion einen einzigen Indexeingang. Wenn man ein 2D-Array mit der gleichen Funktion verbindet, werden zwei Indexeingänge angezeigt: einer für die Zeile und einer für die Spalte. Man kann mit diesen Funktionen auf mehr als ein Element oder Teilarray (Zeile, Spalte oder Seite) zugreifen, indem man die Größe der Funktion mit Hilfe des Positionierwerkzeugs manuell ändert.

Im Blockdiagramm in Abbildung A.14 werden mit Hilfe der Funktion „Array indizieren” eine Zeile und ein Element aus einem 2D-Array entnommen.

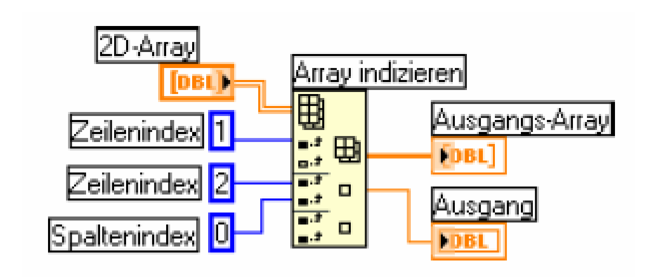


Abbildung A.14: Entnahme von Daten aus einem 2D-Array durch die Funktion „Array indizieren"

Um auf mehrere aufeinander folgende Werte in einem Array zuzugreifen, erweitert man die Funktion „Array indizieren" um drei Felder, verbindet jedoch mit den Index-Eingängen keine Werte. Um beispielsweise die erste, zweite und dritte Zeile eines 2D-Arrays zu extrahieren, erweitert man die Funktion um drei Felder und verbindet mit jedem Teilarray-Ausgang 1D-Array-Anzeigeelemente.

Kapitel 8 – Graphen und Diagramme

Mit Graphen und Diagrammen können Daten in grafischer Form dargestellt werden. Graphen und Diagramme unterscheiden sich hinsichtlich der Anzeige und der Aktualisierung der Daten. VIs mit Graphen erfassen die Daten normalerweise in einem Array und stellen die Daten anschließend in einem Graphen dar, ähnlich wie in einer Tabellenkalkulation, bei der die Daten zuerst gespeichert und dann grafisch dargestellt werden. Bei einem Diagramm werden dagegen zu den bereits in der Anzeige vorhandenen Datenpunkten neue hinzugefügt, so dass eine Daten-Historie erzeugt wird. Deshalb ist es möglich, in Diagrammen die aktuellen Messwerte bzw. die aktuelle Messung in Zusammenhang mit den zuvor erfassten Werten anzuzeigen.

Es gibt folgende Arten von Graphen und Diagrammen:

- Kurvengraph und Kurvendiagramm: zeigen die erfassten Daten mit konstanter Rate an
- XY-Graph: zeigt die erfassten Daten mit nicht konstanter Rate an, wie z. B. Daten, die getriggert erfasst werden

Anpassen der Darstellung von Graphen und Diagrammen

Die Darstellung von Graphen und Diagrammen kann durch Ein- oder Ausblenden von Optionen angepasst werden. Dazu klickt man mit der rechten Maustaste auf den entsprechenden Graphen oder das Diagramm und wählt aus dem Kontextmenü „Sichtbare Objekte“, um die folgenden Optionen ein- oder auszublenden:

- Legende der Kurve: legt Farbe und Darstellungsart der Kurve fest. Wenn mehrere Kurven vorhanden sind, zieht man die Legende auf, um alle Kurven anzuzeigen
- Achsenlegende: bestimmt die Achsenbeschriftungen und -eigenschaften.
- Graphenpalette: ändert Skalierung und Formatierung während der Ausführung eines VIs
- x- und y-Achse: dient zur Achsenformatierung
- Cursor-Legende (nur bei Graphen): zeigt an den angegebenen Koordinaten eine Markierung an. Es ist möglich, in einem Graphen mehrere Cursor anzuzeigen
- Bildlaufleiste: dient zum Scrollen in einem Graphen oder Diagramm

Automatische Skalierung

Die Achsen von Graphen können automatisch an die darzustellende Kurve angepasst werden. Ein solchen Verhalten wird als automatische Skalierung bezeichnet. Um die automatische Skalierung zu (de-)aktivieren, klickt man mit der rechten Maustaste auf den Graphen und wählt aus dem Kontextmenü „x-Achse » Autom. Skalierung x- oder y-Achse » Autom. Skalierung y“ aus. Standardmäßig ist die automatische Skalierung für Graphen aktiviert.

Achsenlegende für Kurvengraphen

Über die Achsenlegende sind Beschriftung und Eigenschaften der Achsen einstellbar. Man klickt unter Verwendung des Bedienwerkzeugs auf die links angezeigte Schaltfläche „Achsenstil“, um Format, Genauigkeit und Abbildungsmodus zu konfigurieren. Über die links dargestellte Schaltfläche „Skalierungssperre“ kann die automatische Skalierung für jede Achse einzeln aktiviert bzw. deaktiviert werden.

Achsenformatierung

Um die Darstellungsart der x- und y-Achse eines Graphen oder Diagramms festzulegen, wählt man im Dialogfeld „Signalverlaufs-Eigenschaften“ bzw. „Diagramm-Eigenschaften“ die Seite Format und Genauigkeit aus. Als Standardbeschriftung wird für die x-Achse „Zeit“ und für die y-Achse „Amplitude“ verwendet. Die Werte der x-Achse sind auf Fließkommadarstellung voreingestellt, die der y-Achse passen sich automatisch an die Eingangswerte des Graphen an. Um die Konfiguration der Achsen zu ändern, klickt man den Graphen oder das Diagramm mit der rechten Maustaste an und wählt aus dem Kontextmenü die Option „Eigenschaften“.

Das Zahlenformat für die Werte der Achsen kann im Dialogfeld „Signalverlaufs-Eigenschaften“ bzw. „Diagramm-Eigenschaften“ unter „Format und Genauigkeit“ ausgewählt werden. Unter „Skalierungen“ lassen sich Achsenbeschriftung und Skaleneinteilung verändern. Die Werte an der Achse sind standardmäßig so konfiguriert, dass ab der siebten Stelle automatisch in die Exponentialschreibweise umgeschaltet wird.

Kurven-Graphen

In Kurvengraphen werden in gleichen Abständen abgetastete Messungen angezeigt (Abbildung A.15):

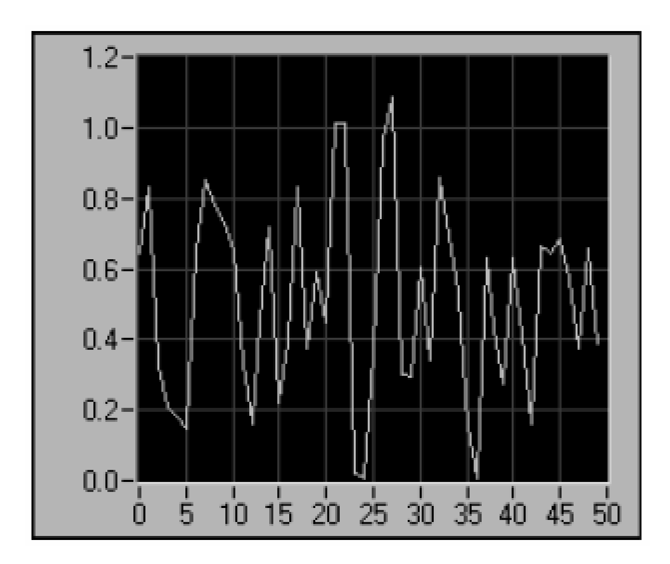


Abbildung A.15: Beispiel eines Kurvengraphens

Der Kurvengraph stellt nur einwertige Funktionen dar, wie zum Beispiel bei erfassten zeitabhängigen Signalen. Er kann Kurven mit einer beliebigen Anzahl von Werten anzeigen und arbeitet mit mehreren Datentypen, was den Zeitaufwand gering hält, da die Daten vor der Anzeige nicht umgewandelt werden müssen.

Bei einem Kurvendiagramm handelt es sich um einen speziellen Typ eines numerischen Anzeigeelements, das eine oder mehrere Kurven anzeigt. Wenn an ein Diagramm ein Einzelwert oder mehrere Werte zur gleichen Zeit übergeben werden, interpretiert LabVIEW diese als Punkte des Diagramms und erhöht den x-Index beginnend bei $x = 0$ jeweils um eins, da die Daten als neue Werte für eine Einzelkurve verstanden werden. Bei Signalverlaufsdaten passt sich der x-Index an das angegebene Zeitformat an. Wie oft das Diagramm neu gezeichnet wird, hängt davon ab, wie oft Daten übergeben werden. Wenn man mehrere Werte für Kurven in einem einzigen Aktualisierungsvorgang übergeben möchte, verbindet man ein Array mit dem Diagramm. Jeder Zahlenwert stellt einen einzelnen y-Wert für die einzelnen Kurven dar. Wenn die Anzahl der anzuzeigenden Kurven erst zur Laufzeit ermittelt werden kann oder wenn in einem einzigen Aktualisierungsvorgang mehrere Werte für verschiedene Kurven übergeben werden sollen, verbindet man das Diagramm mit einem 2D-Array mit numerischen Werten. Wie beim Kurvengraph interpretieren Kurvendiagramme die Zeilen standardmäßig als neue Daten für die einzelnen Kurven.

Kapitel 9 – Daten I/O

Datei-I/O

Bei Datei-I/O-Operationen werden Daten aus Dateien ausgelesen oder in Dateien geschrieben. Die VIs zur Datei-I/O eignen sich für alle mit der Datei-I/O in Verbindung stehenden Vorgänge, wie beispielsweise:

- Öffnen und Schließen von Dateien.
- Auslesen von Daten und Schreiben von Daten in Dateien
- Gleiches im Tabellenkalkulationsformat
- Verschieben und Umbenennen von Dateien und Verzeichnissen
- Ändern von Dateieigenschaften
- Erstellen, Ändern und Lesen einer Konfigurationsdatei

Auswahl eines Datei-I/O-Formats - Einsatz von Textdateien

Dateien im Textformat sollten immer dann verwendet werden, wenn die Daten anderen Anwendern oder Applikationen zur Verfügung gestellt werden sollen, kein wahlfreier Schreib- bzw. Lesezugriff erfolgen soll oder numerische Genauigkeit, Festplattenspeicherplatz und Geschwindigkeit der Datei-I/O nicht von Bedeutung sind. Für eine gemeinsame Nutzung sind Textdateien am besten geeignet. Sie können von nahezu jedem Computer gelesen und geschrieben werden.

Erstellen von Text- und Tabellenkalkulationsdateien

Beim Schreiben von Text in Textdateien ist keine Formatierung notwendig, da eine solche für die meisten Textverarbeitungsprogramme nicht erforderlich ist. Mit dem VI „Zeichen in Datei schreiben" können Text-Strings in eine Textdatei geschrieben werden. Das VI öffnet und schließt die Datei automatisch. Mit dem VI „In Spreadsheet-Datei schreiben" oder der Funktion „Array in Tabellen-String" können Zahlen aus einem Graphen, einem Diagramm oder Werte von einer Erfassung in einen Tabellen-String konvertiert werden.

Zur Übergabe von Zahlen oder Text an Tabellenkalkulations- oder Textverarbeitungsprogramme sollten zum Formatieren und Zusammenfassen der Daten die String- und Array-Funktionen verwendet werden. Anschließend sind die Daten in eine Datei zu schreiben.

Formatieren und Schreiben von Daten in Dateien

Mit der Funktion „In Datei formatieren" können String-, Pfad-, numerische und boolesche Daten als Text formatiert und dieser Text in eine Datei geschrieben werden.